Comparing Numeric Stability of Euler angles and Quaternions when Performing Rotations

Daniel Smith (11306026)

April 10, 2014

1 Introduction

Euler angles have long been used to perform rotations in three-dimensional (3D) space, in both the fields of science and engineering. Euler angles are represented as three values: pitch, yaw, and roll. These values represent rotations about the X, Y, and Z axes respectively. Euler angles are simple to understand, implement, use, and to visualize. However, they suffer from a horrible affliction known as gimbal lock, where one can lose one or more degrees of freedom. To make matters worse, Euler angles supposedly suffer from numeric instability when calculated on a computer using limited-precision floating point mathematics.

Quaternion Mathematics, an amazing invention by a man named Hamilton, is a newer way of representing rotations in 3D space. They remove the problem of gimbal lock, are reported to suffer much less from numeric instability, and add a number of new features such as linear interpolation of rotations. Rotations using quaternions are even faster to compute than when using euler angles. However, despite it being the holy grail of 3D rotations, it comes at the cost of being difficult to understand and visualize. However, there are numerous maths libraries, including the well-known GLM library, that include complete implementations of quaternions, making their use somewhat simpler.

In this report I will investigate and compare the numeric instability of Euler angles and quaternions.

2 The Setup

For my experiment I set up a basic OpenGL scene that included a camera and a checkerboard floating in space. I then created two different implementations of the camera: one using Euler angles for rotation, and one using quaternions. I then created two instances of each camera implementation: one using single-precision and the other using double-precision floating point values. I then hooked input handling to all the cameras and drew the scene from each of their viewpoints in a window split into four.

3 The Experiment

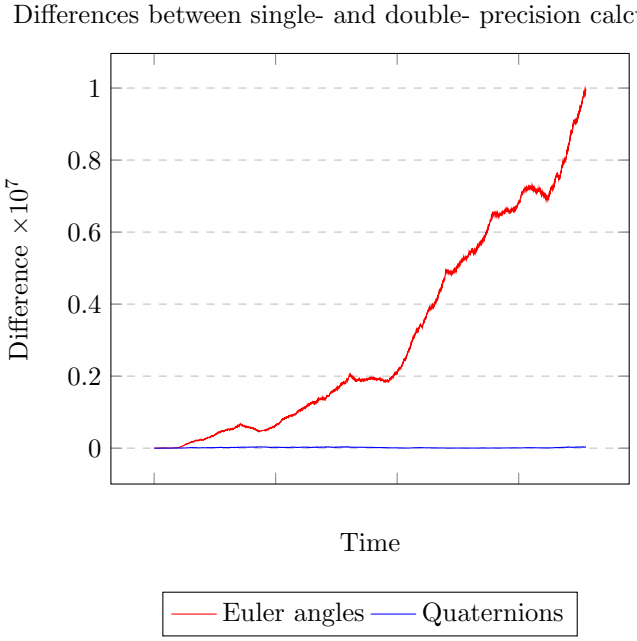
To test for numeric instability, I calculate all rotations both with single-precision and double-precision floating point arithmetic. Due to rounding errors, the

difference between the single- and double-precision calculations should slowly increase. The more numeric instability, the more rounding errors, and ultimately the larger the difference between the two difference calculations.

To observe how each rotation implementation fares, I rotate the cameras around on all axes over a period of time, recording the average difference between the results of each calculating each rotation. Since the rounding errors are compounded, the error should increase over time. The implementation with the most numeric stability, and therefore the least amount of rounding errors, should exhibit the smaller increase in error over time.

4 Results

The results were collected over a few minutes and are shown in the graph below.



5 Conclusion

As can be seen from the results, the amount of accumulated error over time when using Euler angles is huge compared to that of quaternions. In fact, it appears as if quaternions exhibit almost no numeric instability. While I cannot say for sure, it would appear that rounding errors add up exponentially. If that is the case, and this is recorded from only a few minutes, imagine the error that would accumulate over the period of a few hours, or even days.

The solid conclusion is that Euler angles exhibit notably more numeric instability than quaternions do.